Implementation of the chimera method in the unstructured hybrid DLR finite volume Tau-Code

Aziz Madrane, Ralf Heinrich and Thomas Gerhold
German Aerospace Center, DLR
Institute of Aerodynamics and Flow Technology
Lilienthalplatz 7, 38108 Braunschweig
Germany

Abstract

The purpose of this paper is to present a chimera method on unstructured hybrid grids which is implemented in the DLR Tau–code. The chimera technique is assumed to be an efficient way to simulate bodies in relative motion, like control surface deflections.
This paper describes the implementation of a chimera technique capable for efficient and accurate simulation of moving bodies. The technique has been verified for the inviscid and viscous flow around a wing with a flap, moving NACA0012 airfoils and a supersonic 3D ramp.
The presented chimera calculations are in good agreement with single mesh computations performed for comparison. The described developments enable for the application of the DLR Tau–code to a complex test case with industrial relevance including moving flaps and coupling to flight mechanics , which is demonstrated, finally.

Nomenclature

c	Sound speed, [m/s]
M	Mach number
p	Pressure, [Pa]
Re	Reynolds number

V	Velocity, [m/s]
x, y, z	Cartesian body axes, [m]
α	Angle of attack, [deg]
ρ	Density, [kg/m^3]

I. Introduction

Advances in CFD methods enable simulations of non-linear complex flowfieds around moving bodies, and can be a complement to experimental approches. However, for unstructured hybrid grid based flow solvers, a major obstacle is the difficulty in building grids around moving complex geometries. One way to alleviate this problem is to use unstructured composite grids. The main unstructured composite grid techniques emphasised in literature are the non-matching grids and overlapping grids approches [5, 7, 9].

The chimera grid technique belongs to this family. It is an overset grid approch in which grids are generated independently around each body. Moreover, meshes can be moved with respect to each other without any remeshing. This technique seems particularly suited for moving bodies.

The purpose of this paper is to present an implementation of chimera method on unstructured hybrid grids developed at the German Aerospace Center (DLR). It is described in the following sections.

Computational results for a NACA0012 airfoil, a supersonic 3D ramp and a wing with a flap using the Chimera technique are compared with results for a single grid in subsections A,B and C of section III.

Finally, the ability to handle industrial applications including moving control surfaces and coupling to flight mechanics is demonstrated in subsections D and E of section III.

II. Principles of the Chimera method

The principle of the chimera technique is the following : each body involved in the numerical simulation has its own attached mesh independently generated. Due to the arbitrary position of the different grids, some nodes can be located in the solid regions, and so have to be removed from the flow calculation. Therefore chimera meshes introduce holes and artificial boundaries inside the computational domain. Communications between grids are established through artificial boundaries where appropriate values are interpolated from one mesh to another (see Figure 1). There are three steps to establish communications in the overset method:

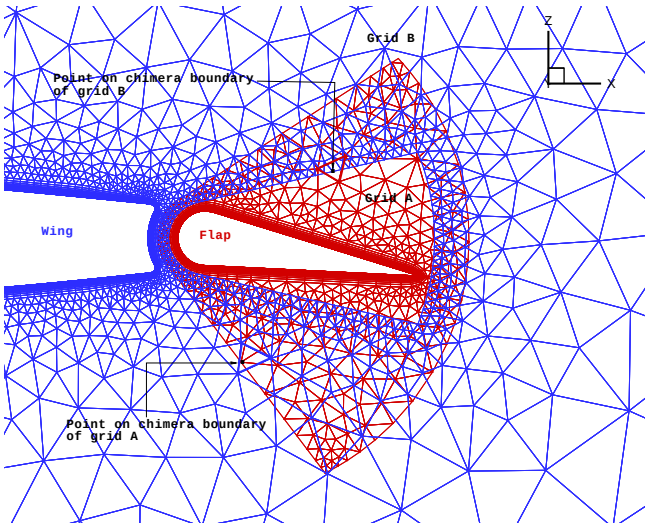

Figure 1: Overset unstructured hybrid grids and definition of the boundary chimera overlap.

1) Definition of grid hierarchy, see subsection A.

2) Hole-cutting : For the cases presented in this paper, we use analytic shapes, as hole-cutting geometry.

3) Identification of interpolation stencils, which involves a search of donor cells for all chimera boundary points, see subsections B and C.

A. Grid hierarchy

The grid hierarchical form we have used for the computations in this article is similar to the one used by Benek et al [2]. The entire region is covered by one main grid, called G_1. This grid contains all the subgrids G_k where $k = 2, ..., N$ is the global ordering of the grids, to be defined by the user. The more important a grid is, the higher number it will have in the sequence, i.e. usually finer grids have a higher number (see Figure 2). Each subgrid G_k is completely contained in one or more underlying grid G_m, where $m < k$. The highest numbered underlying grids G_m is called the local main grid of the subgrid G_k. For example

G_1 is the local main grid of $G_2, G_3,$ and G_4.

G_2 is the local main grid of G_5.

G_4 is the local main grid of G_6.

G_5 is the local main grid of G_7.

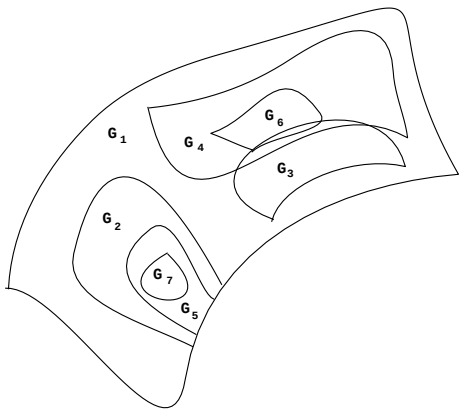

Figure 2: Grid hierarchy.

B. Search Algorithms:

Neighbor-to-Neighbor search (N2N)

Consider using the cell-based data structure with given vertices and neighbors (see Figure 3). Starting from a given cell, we traverse to the neighbor that lies in the direction of the target, until we have reached the cell that contains the target. The direction can be determined by calculating the scalar product of vector from midside of each edge with the outward normal on that edge (see Figure 3). Only for the cell that contains the target, all scalar products are negative. This method has been modified for the edge-based data structure (see Figure 4) of the Tau-code [4].

Consider the interpolation point t and an arbitrary point p_i of the grid G_k, the element is searched for which the target belongs to. Then all elements surrounding point p_i (see Figure 4) are checked first. If point t doesn't fall into one of these elements, the edge connected to point p_i, that encloses the smallest angle with vector $\overrightarrow{p_i t}$ is determined. We take the other point of this edge as new starting point and run the algorithm again, until the desired element is found. This algorithm might fail, if the domain of computation is not convex (see Figure 5). This is avoided by using a brute force algorithm [6].

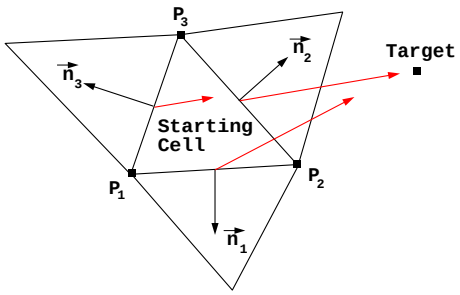

Figure 3: A cell-based datastructure with pointers to the three neighboring cells for each cell, and neighbor to neighbor search method.

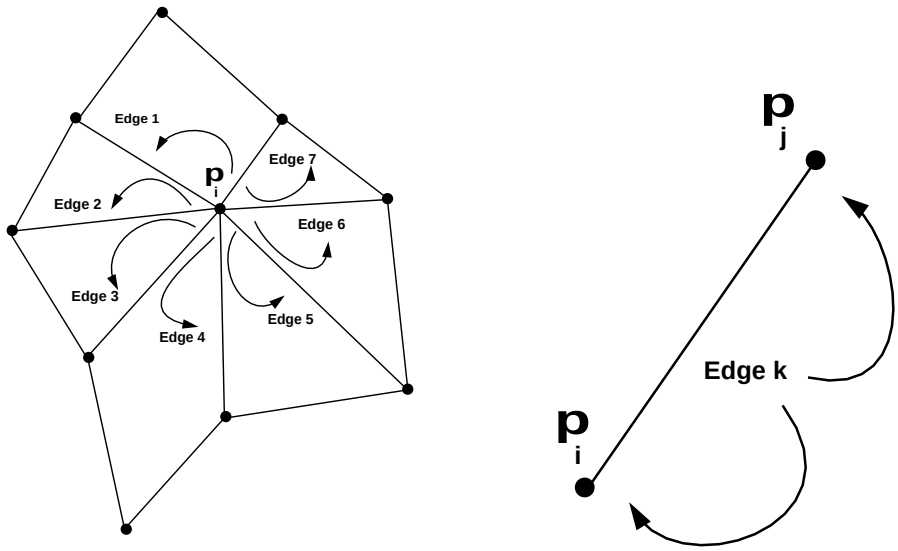

Figure 4: Two edge-based datastructures with vertex-to-edge pointers (left) and edge-to-vertex pointers (right)

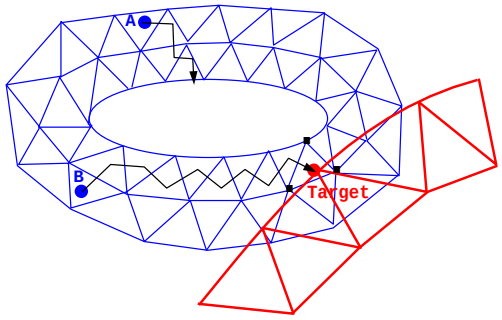

Figure 5: The neighbor to neighbor search method. The alghorithm might fail when the domain of computation is not convex

Alternating Digital Tree (ADT)

Additionally we have implemented an ADT [3] search algorithm, because N2N is time consuming and too expensive for non–convex domains. The steps involved in generating an ADT and searching are outlined as follows:

1^{st}step: Determine the bounding boxes $(\{x_{min}, y_{min}, z_{min}\}, \{x_{max}, y_{max}, z_{max}\})$ (Figure 6) of all the elements in each grid of the overset mesh system.

2^{nd}step: Build the ADT for each grid, the dataset used for building the tree comprises the bounding box coordinates of the grid cells.

3^{rd}step: Search in the tree for a given target point. The algorithm searches the ADT of the given grid and creates a list of cells whose bounding boxes overlap for a given target point I_B (see Figure 7). To identify the cell that contains the target point, the following criterion

$$min(N_i, 1 - N_i) \geq 0, \quad \forall i \tag{II.1}$$

must be satisfied. N_i are the shape functions well known from finite element methods [1]. They are computed using the formula $N_i(P_j) = \delta_{ij}$, where δ_{ij} is the Kronecker's symbol. Other types of elements are split into tetrahedrons and the shape functions are computed for each of the subelements. Once a donor cell is identified, linearly interpolate the solution at the nodes of the donor cell e_A to the target point I_B (see figure 7) as follows :

$$f_{I_B} = \sum_{i_A} f_{i_A} \cdot [N_{i_A}]_{I_B} \quad \text{with} \quad \sum_{i_A} N_{i_A} = 1 \tag{II.2}$$

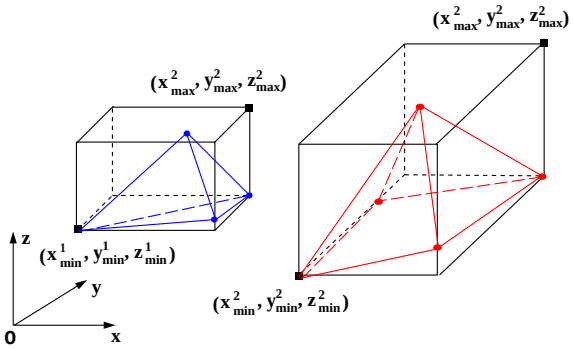

Figure 6: Cartesian bounding boxes of tetrahedra and pyramid.

where f_{i_A} represents the solution at the corner nodes of the element e_A and $[N_{i_A}]_{I_B}$ represents the shape function of the element e_A evaluated at the target point I_B.

C. Calculations of shape functions.

Linear case :

The interpolation is based on linear shape functions well known from finite element methods. For tetrahedral cells, the linear shape functions are given by the following formulas; $N_1 = 1 - \xi - \eta - \zeta$, $N_2 = \xi$, $N_3 = \eta$ and $N_4 = \zeta$, where (ξ, η, ζ) represent the coordinates in the parametric space (see Figure 8). For other types of elements, hexahedron, prism and pyramid , the interpolation is also based on linear shape functions by splitting those elements into tetrahedrons (see Figures 9, 10 and 11) . Table 1 gives all possible subdivisions of the hexahedron p_1 to p_8 into five or six tetrahedra as a function of the number n of diagonals through the vertex p_7.

Nonlinear case :

This subsection describes a method developed for function interpolation within different element types, which is based on the finite element interpolation theory.

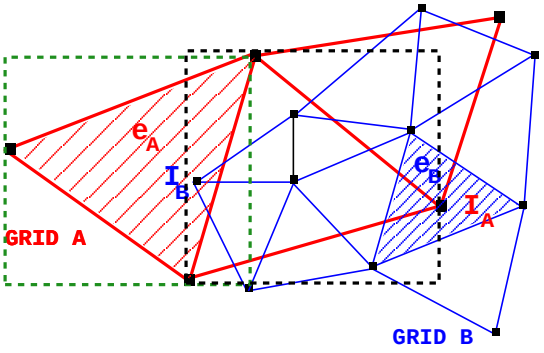

Figure 7: Overlapping bounding boxes.

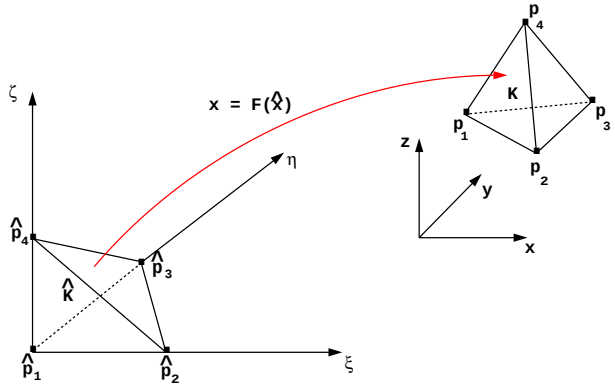

Figure 8: Parametric space.

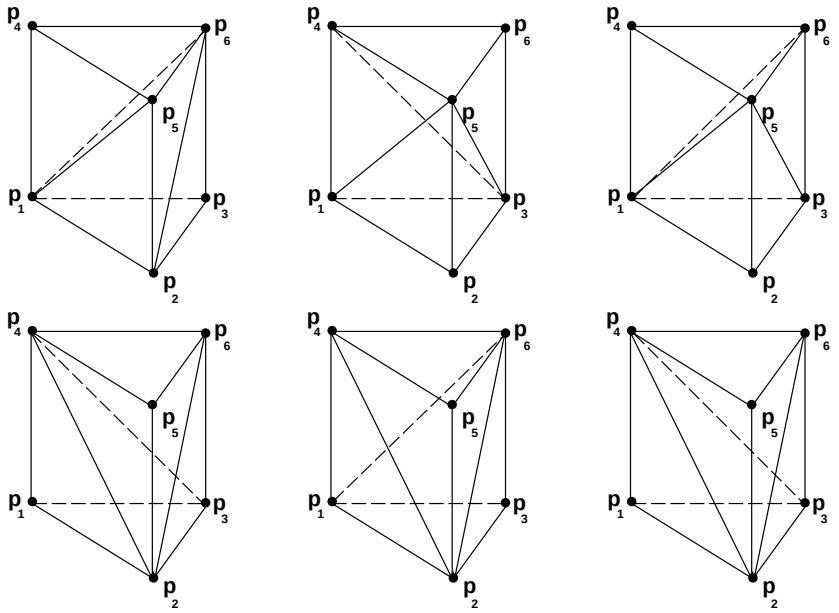

Figure 9: The six ways to split the quadrilateral faces of the prism such that it can be subdivided into three tetrahedra.

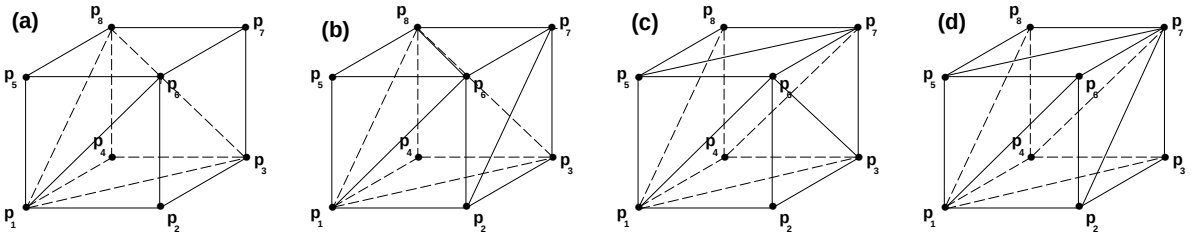

Figure 10: First configuration with no diagonal that goes through vertex p_7 (a), second configuration with one diagonal that goes through vertex p_7 (b), third configuration with two diagonals that go through vertex p_7 (c) and fourth configuration with three diagonals that go through vertex p_7 (d)

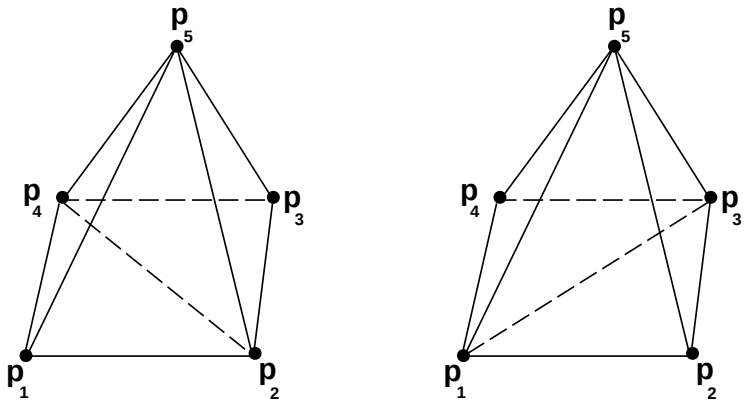

Figure 11: The two ways to split the quadrilateral face of the pyramid and to subdivide the pyramid into two tetrahedra.

n	$\triangle_1$	$\triangle_2$	$\triangle_3$	$\triangle_4$	$\triangle_5$	$\triangle_6$
0	$1,2,3,6$	$1,3,8,6$	$1,3,4,8$	$1,6,8,5$	$3,8,6,7$	Nil
1	$1,6,8,5$	$1,2,8,6$	$2,7,8,6$	$1,8,3,4$	$1,8,2,3$	$2,8,7,3$
	$1,6,8,5$	$1,2,7,6$	$1,7,8,6$	$1,8,3,4$	$1,8,7,3$	$2,1,7,3$
2	$1,5,6,7$	$1,4,8,7$	$1,8,5,7$	$1,2,3,6$	$1,4,7,3$	$1,7,6,3$
	$1,3,4,7$	$1,5,7,8$	$1,7,4,8$	$1,2,3,6$	$1,7,5,6$	$1,3,7,6$
3	$1,3,4,7$	$1,4,8,7$	$1,8,5,7$	$1,6,7,5$	$2,6,7,1$	$2,7,3,1$
	$1,2,7,6$	$1,7,8,5$	$1,6,7,5$	$1,7,2,3$	$1,8,7,4$	$1,7,3,4$
	$1,2,7,6$	$1,2,3,7$	$1,3,4,7$	$1,6,7,5$	$1,7,4,8$	$1,7,8,5$

Table 1: All possible subdivision of the hexahedrons into six tetrahedrons.

Following this theory, a function F is approximated as :

$$F(\xi,\eta,\zeta) = \sum_i N_i(\xi,\eta,\zeta)F_i$$

where (ξ,η,ζ) represent the coordinates in the parametric space, N_i the shape functions associated to the node i of the element and F_i the value of the function at this node.

Let denote $H = (P_1,P_2,P_3,P_4,P_5,P_6,P_7,P_8)$ denote a hexahedron, and M a point, we obtain $F(M)$ by the following formula:

$$F(M) = \sum_i^8 N_i(\xi_M,\eta_M,\zeta_M)F_i \tag{II.3}$$

where

$$N_1 = 0.125(1-\xi)(1-\eta)(1-\zeta), \quad N_2 = 0.125(1+\xi)(1-\eta)(1-\zeta)$$

$$N_3 = 0.125(1+\xi)(1+\eta)(1-\zeta), \quad N_4 = 0.125(1-\xi)(1+\eta)(1-\zeta)$$
$$N_5 = 0.125(1-\xi)(1-\eta)(1+\zeta), \quad N_6 = 0.125(1+\xi)(1-\eta)(1+\zeta)$$
$$N_7 = 0.125(1+\xi)(1+\eta)(1+\zeta), \quad N_8 = 0.125(1-\xi)(1+\eta)(1+\zeta)$$

To determine the coefficients (ξ_M, η_M, ζ_M), we set $F(M) = \overrightarrow{P_1M}$ to (II.3), and we obtain:

$$\begin{aligned} \overrightarrow{P_1M} &= N_2\overrightarrow{P_1P_2} + N_3\overrightarrow{P_1P_3} + N_4\overrightarrow{P_1P_4} + N_5\overrightarrow{P_1P_5} + \\ &\quad N_6\overrightarrow{P_1P_6} + N_7\overrightarrow{P_1P_7} + N_8\overrightarrow{P_1P_8} \end{aligned} \tag{II.4}$$

Using (II.4), the following function can be built:

$$\begin{aligned} G(\xi_M, \eta_M, \zeta_M) &= N_2\overrightarrow{P_1P_2} + N_3\overrightarrow{P_1P_3} + N_4\overrightarrow{P_1P_4} + \\ &\quad N_5\overrightarrow{P_1P_5} + N_6\overrightarrow{P_1P_6} + N_7\overrightarrow{P_1P_7} + \\ &\quad N_8\overrightarrow{P_1P_8} - \overrightarrow{P_1M} \end{aligned} \tag{II.5}$$

The coefficients (ξ_M, η_M, ζ_M) are solution of $G(\xi_M, \eta_M, \zeta_M) = \overrightarrow{0}$, which is solved using Newton method, i.e.

$$\overrightarrow{\xi_M^{n+1}} = \overrightarrow{\xi_M^n} - G_\xi^{-1} G \tag{II.6}$$

where G_ξ is the Jacobian matrix of the function G. Then, the interpolation weights are computed as $N_i(\overrightarrow{\xi_M})$.

A similar procedure is applied to prisms and pyramids, with the corresponding shape functions

$$N_1 = 0.5(1-\zeta)(1-\xi-\eta), \quad N_2 = 0.5(1-\zeta)\xi, \quad N_3 = 0.5(1-\zeta)\eta$$
$$N_4 = 0.5(1+\zeta)(1-\xi-\eta), \quad N_5 = 0.5(1+\zeta)\xi, \quad N_6 = 0.5(1+\zeta)\eta$$

for prisms and

$$N_1 = 0.25[(1-\xi)(1-\eta)-\zeta], \quad N_2 = 0.25[(1+\xi)(1-\eta)-\zeta]$$
$$N_3 = 0.25[(1+\xi)(1+\eta)-\zeta], \quad N_4 = 0.25[(1-\xi)(1+\eta)-\zeta], \quad N_5 = \zeta$$

for pyramids.

III. Numerical results

A. Steady Euler calculation for a supersonic test case (a Ramp)

For the validation of the steady chimera algorithm a simplified supersonic test case is defined to see how the algorithm handles shock waves crossing the chimera boundary, and to compare the results

with an equivalent single block calculation. The setup of the test case is shown in (figure 13). First, a single grid covering the entire flow field was generated. This single grid contains 215947 points and 1211406 tetrahedral cells . For a test of the chimera method two blocks were generated independenly. Block 1 and block 2 contain 119663 and 133577 points, respectively.

The channel is bounded at the bottom and the top by two slip walls. The computations were performed using both the single grid and chimera grid (2 blocks) at a freestream Mach number of 2.5 and an angle of attack of 0^o.

The ramp at the bottom produces an oblique shock that crosses the chimera boundaries. The pressure contours in the cut plane do not show a significant distortion or disappearance of the shock (Figure 13). The figure also shows that the results of both test cases are in excellent agreement. Figure 12 shows the convergence histories of both single and chimera grid solutions, using Roe scheme for the convective flux, with 300 multigrid cycles (4-Level W-type), and ADT for search algorithm with linear interpolation. The chimera boundaries contain 10259 points. Table 2 shows the CPU time for ADT and N2N algorithms. Both solutions have about the same convergence rate indicating that the efficiency of the flow solver is not degraded with the current overlapping strategy.

Time/Interpolation	Linear interpolation	Nonlinear interpolation
Search time (N2N,SGI)	53 sec	39 sec
Search time (ADT,SGI)	26 sec	24 sec

Table 2: The search time needed by the search algorithms for different types of interpolation (3D-Ramp).

B. Steady Euler calculation for a Wing with a Flap

A 2-D wing with a flap was extended in the spanwise direction, as seen in (Figure 14). The flow conditions were given as $M = 0.4$, $\alpha = 0^o$. The trailing edge flap is deflected 20 degrees. We define a chimera boundary around the flap and generate a flap grid composed of 3220 nodes, and a main airfoil grid with 30982 nodes, respectively see (Figure 15) the boundaries chimera contain 480 points. The computation is performed using Roe scheme for the convective flux, ADT for search algorithm with linear interpolation. Within 1000 multigrid cycles (3-Level V-type), the residual fell about five orders. The search time is given in table 3. The computed C_p distribution in the symmetry plane is shown in (Figure 16), and compared with a corresponding one block solution. The results are in excellent agreement.

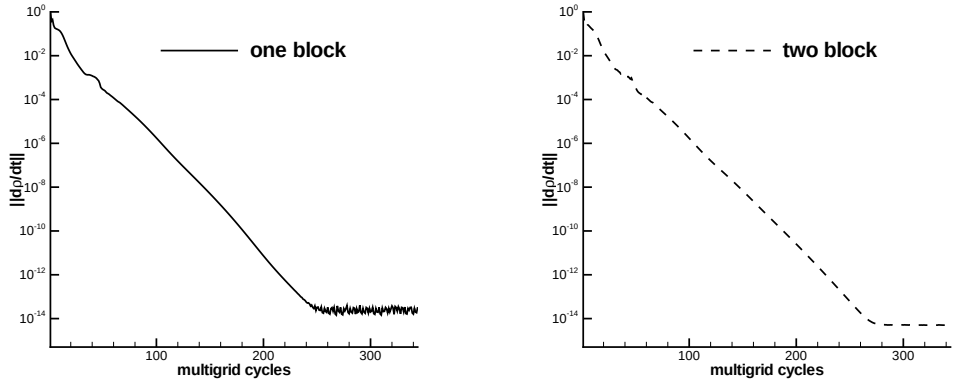

Figure 12: Convergence histories for both single and chimera grids.

Time/Interpolation	Linear interpolation	Nonlinear interpolation
Search time (N2N,SGI)	1.33 sec	0.24 sec
Search time (ADT,SGI)	0.77 sec	0.69 sec

Table 3: The search time needed by the search algorithms for different types of interpolations (Wing-Flap).

C. Unsteady Euler calculation for the NACA0012.

For the verification of the chimera algorithm applied in unsteady computations including moving bodies, a simplified test case of a NACA0012 airfoil with a pitching degree of freedom was chosen, see (Figure 19). Both single and overlapping grids have been used for this computation in order to assess the accuracy and the efficiency of the chimera method. The overlapping mesh used in the computation is shown in (Figure 19), where the first block contains 5125 prisms, 5268 points, the second block (NACA0012) contains 9939 prisms, 10396 points and 200 chimera points . The farfield is fixed in space and time, while the near field is pitching about the center point of the airfoil. The free stream Mach number is 0.755, the pitching oscillation is prescribed by $\alpha = \alpha_o + \Delta\alpha \sin(\omega t)$ with $\alpha_o = 2.51^o$, and a reduced frequency of 0.1628. We performed 50 physical time steps per period of oscillation and three successive periods are computed. Figure 20 shows the C_p contours of the inviscid flow after 100 time steps. One can see the results are in good agreement with one block results (see figures 20, 21). The search time for one physical time step is given in table 4.

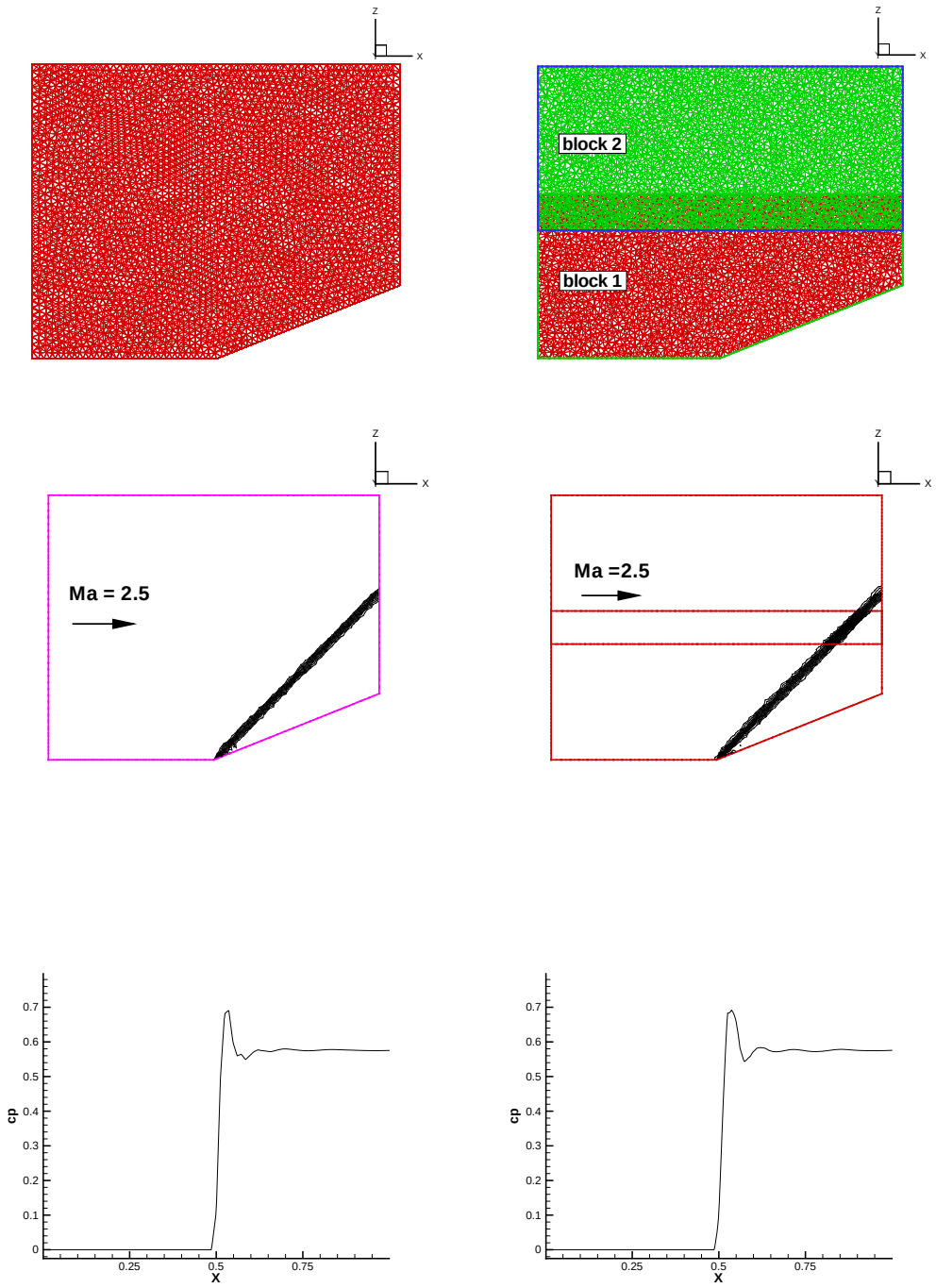

Figure 13: Unstructured grid, pressure contours and C_p distribution on the lower wall for the supersonic flow over a 3D ramp computed on a single grid and chimera grid.

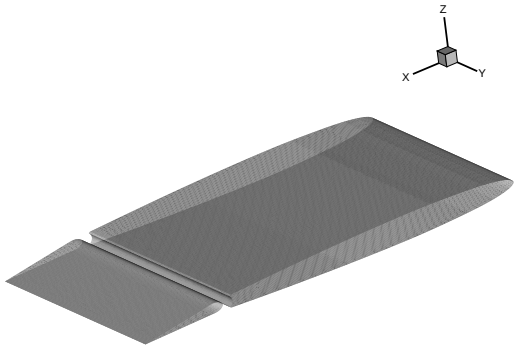

Figure 14: A geometry of wing with a flap.

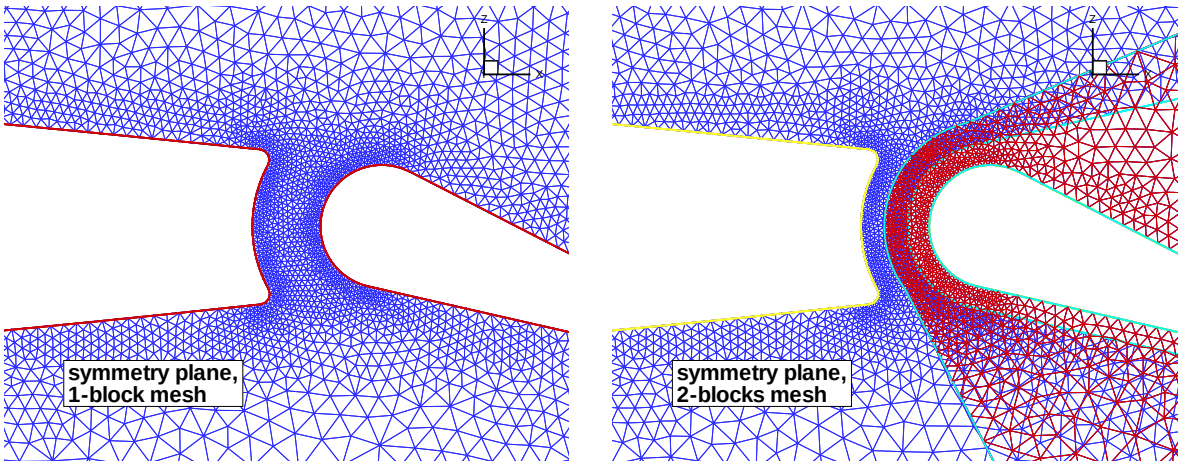

Figure 15: Single and Chimera unstructured mesh used for computing transonic flow over a wing and flap.

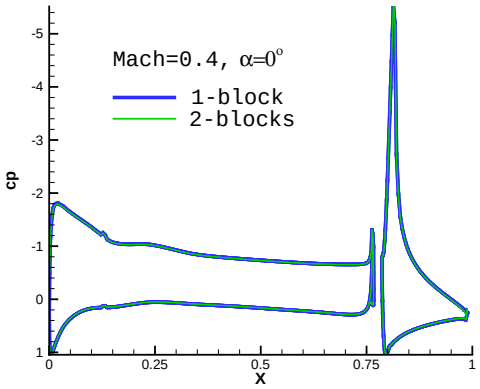

Figure 16: Comparison of C_p distribution between one block and two blocks.

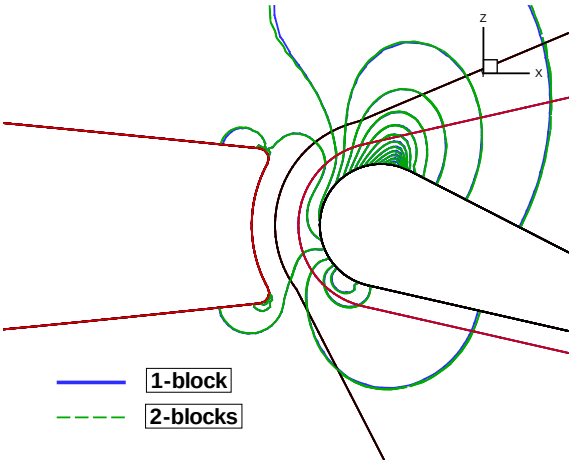

Figure 17: Comparison of isocontours of C_p distribution between one block and two blocks.

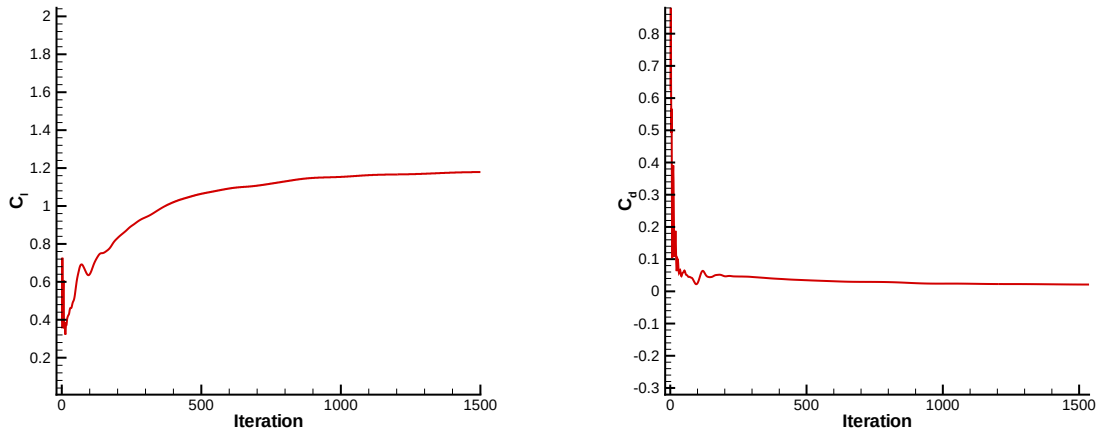

Figure 18: C_l and C_d for two blocks, using Roe scheme.

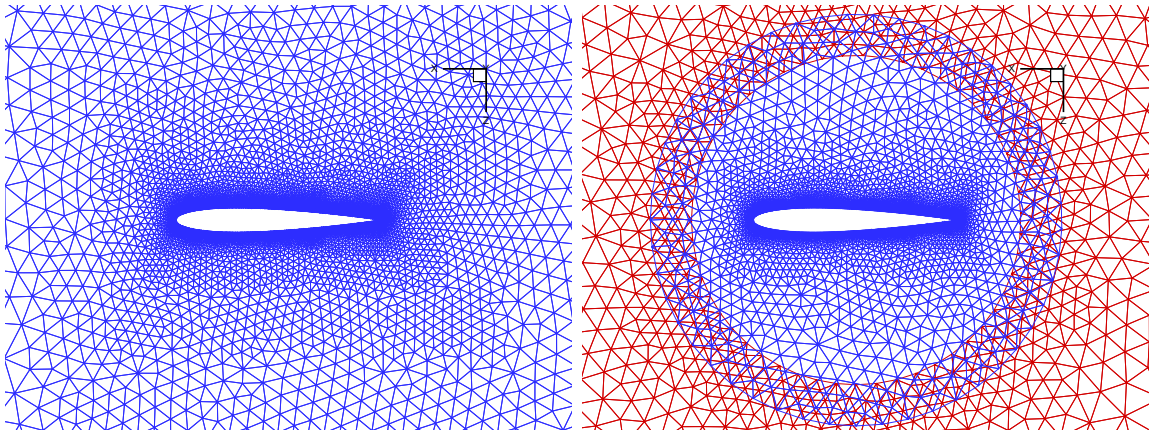

Figure 19: Single and Chimera unstructured mesh used for computing transonic flow over a NACA0012 oirfoil.

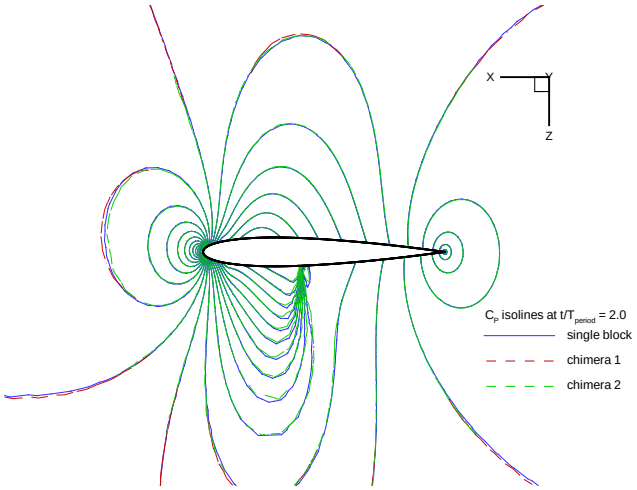

Figure 20: Comparison of C_p contours for inviscid flow between the single grid and overlapping grid solutions for inviscid flow over a pitching NACA0012 airfoil.

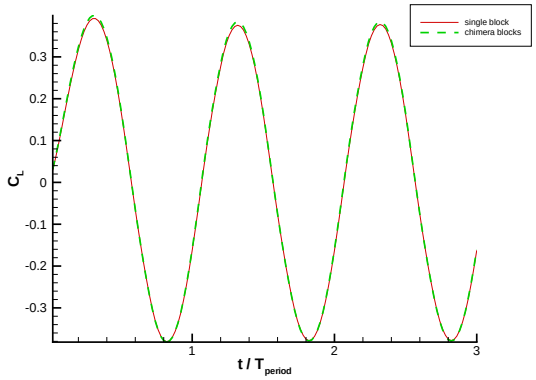

Figure 21: Comparison of C_l coefficient between the single grid and overlapping grid solutions for inviscid flow over a pitching NACA0012 airfoil.

Time/Interpolation	Linear interpolation	Nonlinear interpolation
Search time (N2N,SGI)	0.06 sec	0.06 sec
Search time (ADT,SGI)	0.25 sec	0.23 sec

Table 4: The search time needed by the search algorithms for different types of interpolations (NACA0012).

D. Steady Navier Stokes calculation (delta wing with two flaps)

To evaluate the capability of the overlapping unstructured hybrid grid for multiple moving body problems, the method was applied to the numerical simulation of a delta wing with deflecting flaps, see (Figure 22). For the validation of the numerical simulations, various wind-tunnel experiments were performed in the Transonic Wind Tunnel Göttingen (TWG) [8]. The wind-tunnel model, shown in (Figure 22), is a generic cropped delta-wing aircraft configuration, with fuselage and movable trailing edge flaps. Measurement equipment is installed to determine the aerodynamic forces and moments on the model, as well as the span-wise pressure distribution at two cut-plane locations, at 60 % and 80 % chord length. For the configuration shown in (Figure 22), three

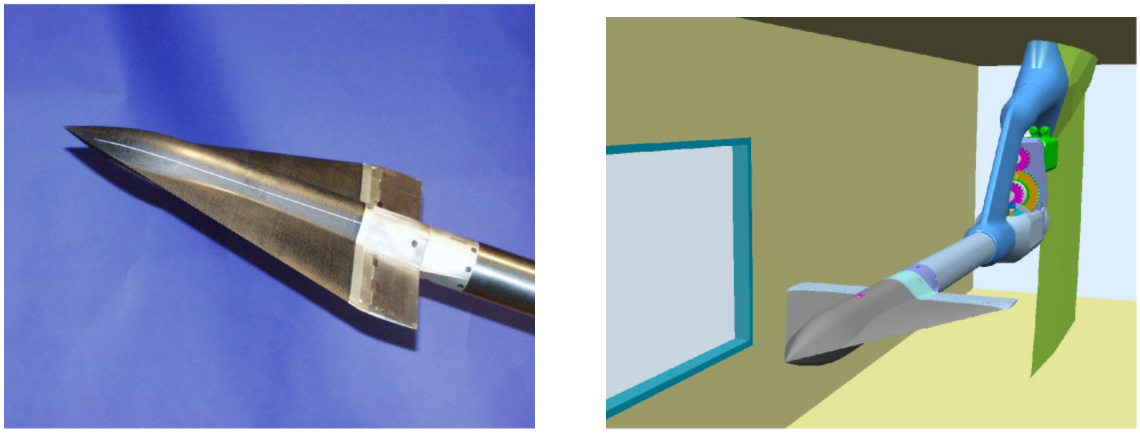

Figure 22: Delta wing with moving flaps.

unstructured grids were generated independently see (Figure 23) which contain 7 million points. Figure 24 shows the detail of the grid through the flap-gap.

The computation was performed for a freestream Mach number of 0.5, angle of attack of 9.3^o and $Re = 3.9$ million. In Figure 25 the calculated surface pressure distribution is shown. The

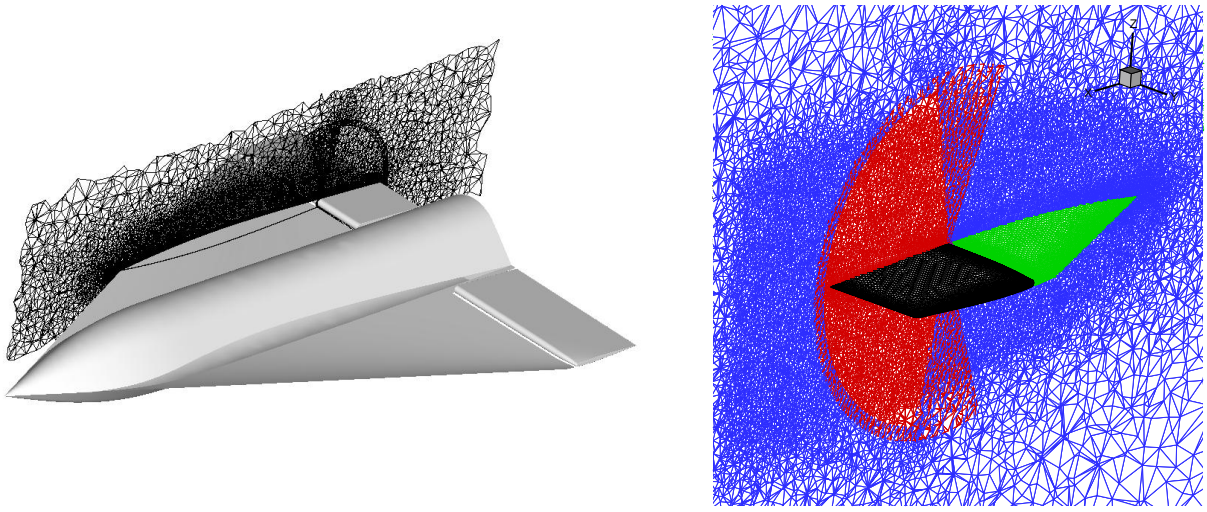

Figure 23: Overlapping unstructured hybrid grids for a delta wing with flaps.

comparison between the calculated and the experimental pressure distribution at 60% and 80% chord length is shown in (Figure 26). The calculated distribution fits well with the experiments at this low pitch angle.

E. Unsteady Navier Stokes calculation (delta wing with two flaps)

Figure 22 shows configurations of a delta wing with two flaps. For this configuration 3 unstructured hybrid grids were generated independently see (Figures 23 and 24). The computation was performed for a freestream Mach number of 0.5, angle of attack of 9.3^o and $Re = 3.9$ million.

In Figures 27 to 31, the results of a free-to-roll simulation initiated by a prescribed flap deflection are shown. The delta wing is released from rest at a roll angle of $\phi = 0^o$, and a constant pitch angle of 9.3^o. The flaps are asymmetrically deflected using the prescribed polynomial shown in figure 27(b), which induces a positive roll-moment on the aircraft and forces it to rotate around its longitudinal axis (a clockwise rotation when looking in the direction of the nose of the aircraft). In figure 31(b) the roll moment is shown over the roll angle. The states of surface pressure distribution

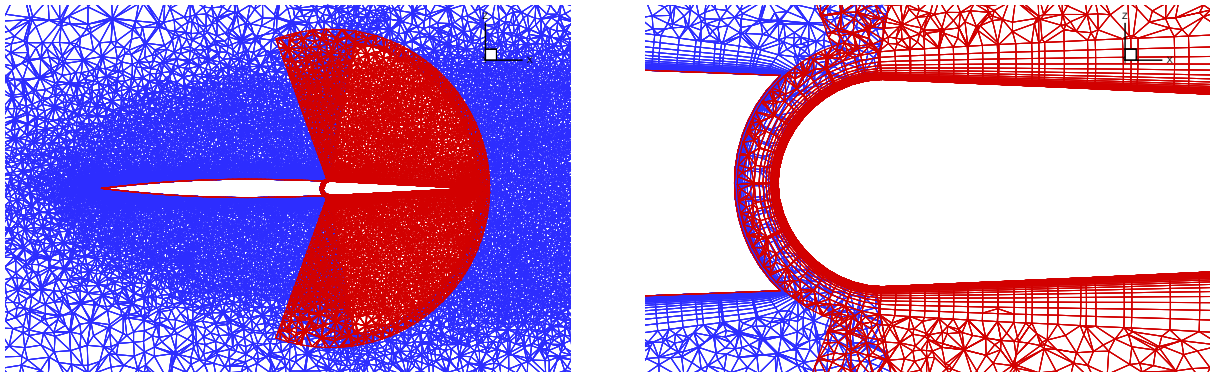

Figure 24: A cut view of the chimera mesh and the detail of the intersection of the grids through the flap-gap.

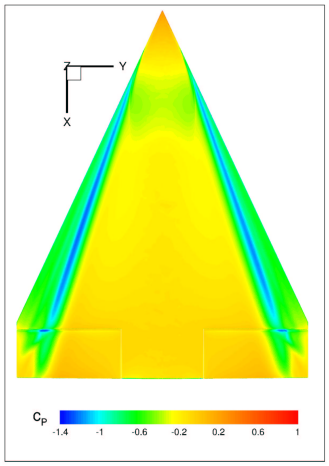

Figure 25: Surface pressure distribution, steady Navier-Stokes calculation (chimera mesh)

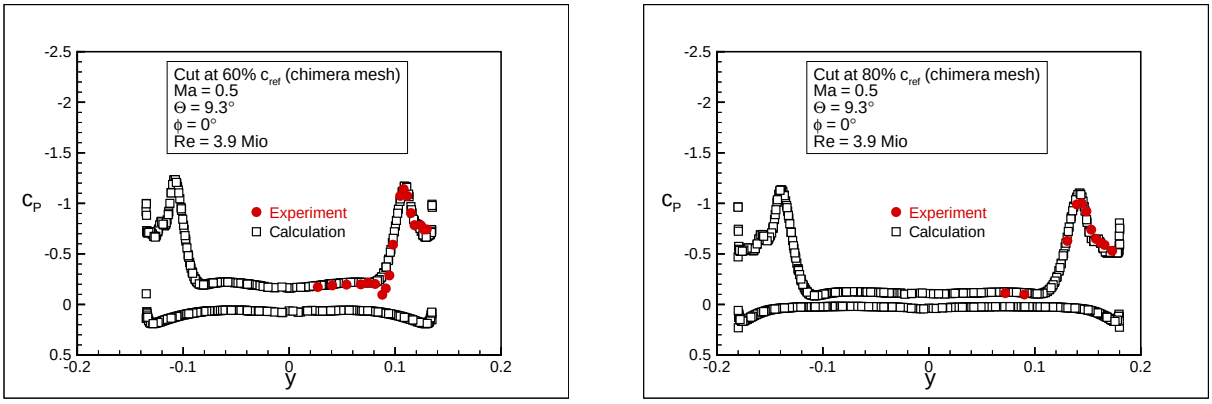

Figure 26: Pressure distribution at 60 % and 80 % chord lenght (steady case, chimera mesh).

shown in figures 27(a) to 30 are specifically marked in figure 31 (b). It is seen that the wing is accelerating up to a roll angle of $\phi = 16^o$. Figure 27(a) shows the surface pressure distribution at the initial state, where the roll moment is zero and the flaps are not deflected. Figure 28 (left) shows the distribution at the first maximum of roll moment, where $\phi = 2.2^o$, and the flaps are half-way deflected at $\eta = 2.5^o$. At $\phi = 16^o$ the roll moment changes sign from positive to negative. The corresponding surface pressure distribution is depicted in figure 28 (right). At this point the flaps are fully deflected at $\eta = 5^o$. From here on the system is damped, and the corresponding surface pressure distributions are shown in figures 28 (right) to 29. As can be seen in these figures, the lower pressure distribution on the luff-side of the wing than on the lee-side causes a negative roll moment and decelerates the wing. At $\phi = 50^o$ the sign of the roll moment changes again, for which the corresponding pressure distribution is shown in figure 29 (right). The wing accelerates once again, but at a lower rate than previously, as can be seen by looking at the gradient of the roll moment in figure 31(b). The corresponding surface pressure distribution, shown in figure 30, also indicates a reduced rate of rotation, as the pressure difference between the luff and lee-side is very slight. It is expected, according to experimental data, that the wing will enter a periodic rotational motion. The calculation would have to be continued to verify that the wing behaves as expected from the experiment.

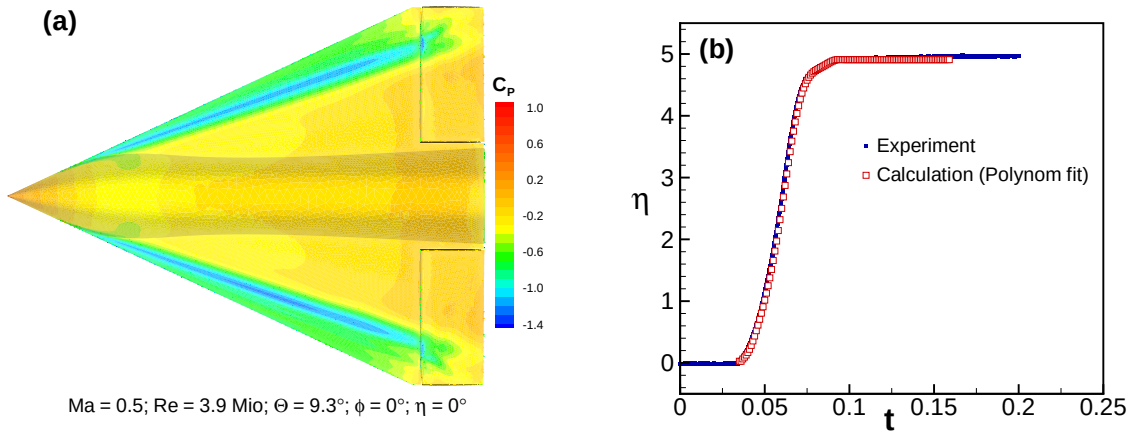

Figure 27: Surface pressure distribution before beginning the deflection of the flaps (a), and flap-deflection angle over time (b).

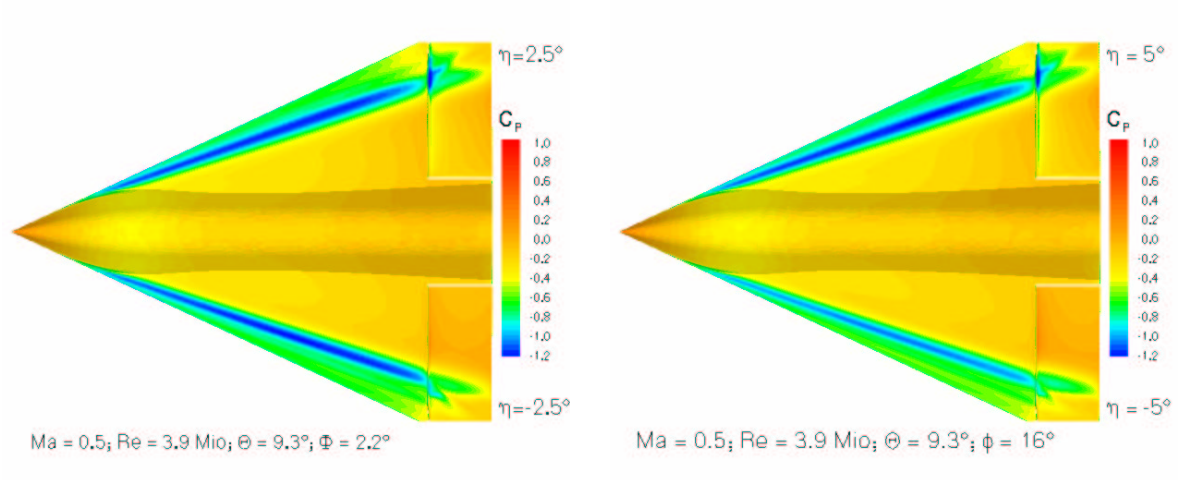

Figure 28: C_p, Free-to-Roll with flap motion.

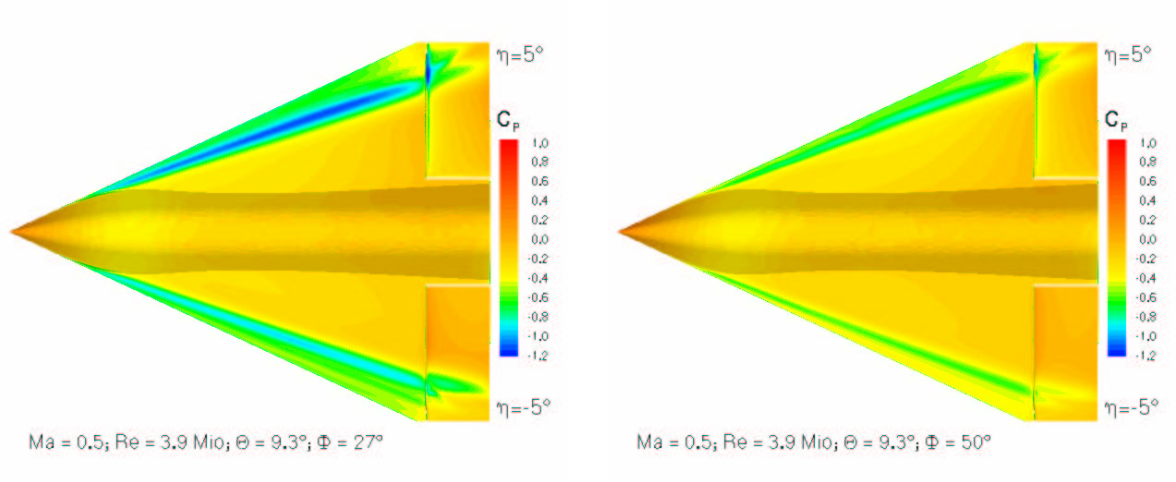

Figure 29: C_p, Free-to-Roll with flap motion.

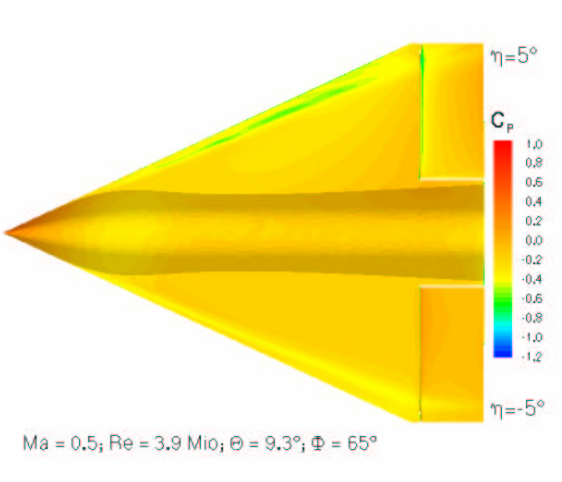

Figure 30: C_p, Free-to-Roll with flap motion.

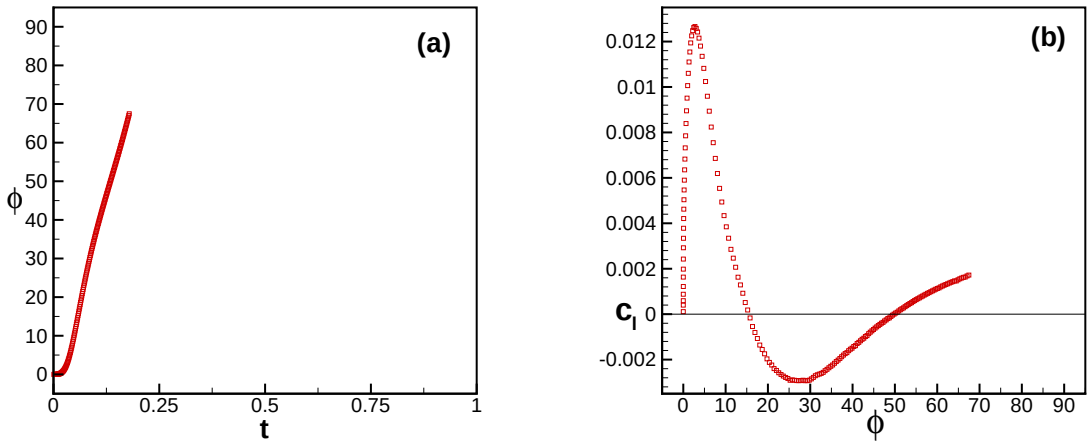

Figure 31: Roll angle over time (a) ; Roll-moment over roll angle (b).

IV. Conclusion

We present an overlapping unstructured hybrid grid method for edge based finite volume codes. The method is accurate and efficient for solving problems with complex time dependent boundaries. We implemented two searchs and two interpolation algorithms, and we have used ADT and nonlinear interpolation for our simulations because they are fast enough for time accurate simulations. We have presented a variety of computations demonstrating the use of the chimera method in the DLR Tau-code.

Future work will concentrate on porting the present methodology to platforms with distributed memory architectures, multiblock adaptation and automatic hole cutting.

V. Acknowledgments

The authors would like to thank A. Schütte for his help in generating the grids for the delta wing, and also wish to thank G. Einarsson for providing us the motion module.

References

[1] I. Babuška, A. K. Aziz. Survey lectures on the mathematical foundations of the finite element method. In the Mathematical Foundation of the Finite Element Method with Applications to Partial Differential Equations, A. K. Aziz, ed., Academic Press, New York - London, pp. 3–363, 1972.

[2] J. A. Benek, P. G. Buning, J. L. Steger. A 3-D Chimera Grid Embedding Technique. Proceeding of the 7th AIAA Computational Fluid Dynamics Conference, Cincinnati, pp. 322–331, 1985.

[3] J. Bonet, J. Peraire. An alternating digital tree (ADT) algorithm for Geometric Searching and Intersection Problems. Int. J. Num. Meth. Eng, Vol. 31, pp. 1–17, 1991.

[4] T. Gerhold, O. Friedrich, J. Evans, M. Galle. Calculation of Complex Three-Dimensional Configurations Employing the DLR–TAU Code, AIAA 97-0167, 1997.

[5] R. Löhner,D. Sharov, H. Luo and R. Ramamurti. Overlapping Unstructured Grids. AIAA Journal, Vol. 1, No. 0439, pp.1–9, march 2001.

[6] R. Löhner. Robust, Vectorized Search Algorithms for Interpolation on Unstructured Grids. Journal of Computational Physics, Vol. 118, pp. 380–387, 1995.

[7] K. Nakahashi, F. Togashi, D. Sharov. Intergrid-Boundary Definition Method for Overset Unstructured Grid Approach. AIAA Journal, Vol 38, No 11, pp. 2077–2084, November 2000.

[8] H. Psolla-Bress, H. Haselmeyer, A. Haddergott, G. Höhler, H. Holst. High-roll rate experiments on a delta wing in transonic flow, ICIAF Record, pp. 369–377.

[9] A. Schütte, G. Einarsson, A. Madrane, B. Schöning, W. Mönnich, R. Kröger. Numerical Simulation of Manoeuvring Aircraft by CFD Aerodynamic and Flight-Mechanic Coupling. RTO Symposium, Paris, April 2002.